CSC 311 – Winter 2022-2023
Design and Analysis of Algorithms

5. Analysis of time efficiency of recursive algorithms
Solving recurrences (Cont.)

Prof. Mohamed Menai
Department of Computer Science
King Saud University

Outline

- Recursion-tree method
- Master theorem

Recursion-tree method

- A recursion tree models the costs (time) of a recursive execution of an algorithm.
- The recursion tree method is good for generating guesses for the substitution method.
- The recursion-tree method can be unreliable, just like any method that uses ellipses (…).
- The recursion-tree method promotes intuition, however.

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

$$T(n)$$

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

$$n^2$$

$$T(n/4) \qquad\qquad T(n/2)$$

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

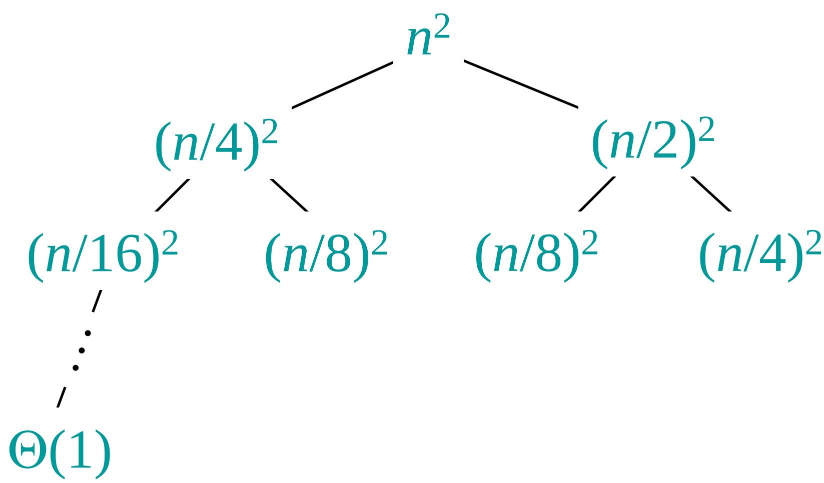
$$n^2$$

$$(n/4)^2 \qquad\qquad (n/2)^2$$

$$T(n/16) \quad T(n/8) \quad T(n/8) \quad T(n/4)$$

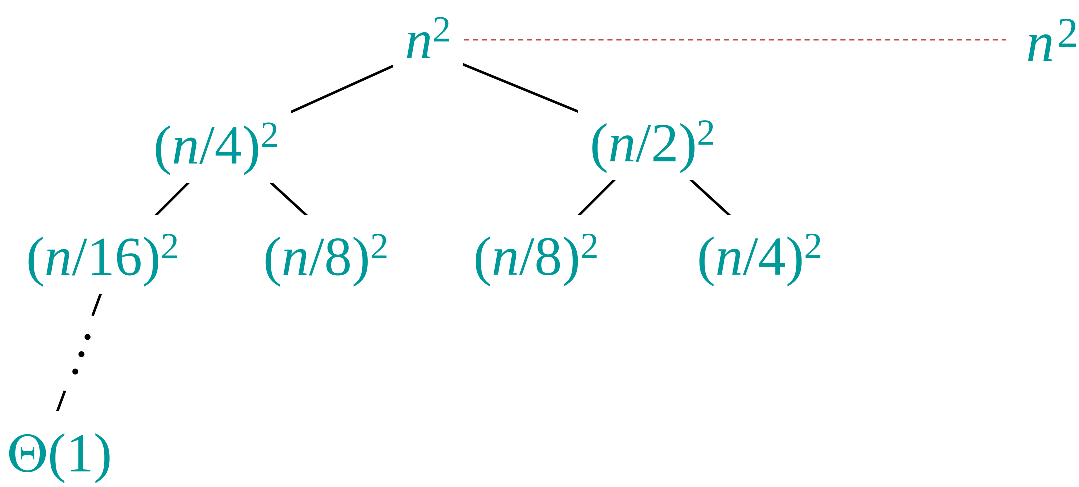
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



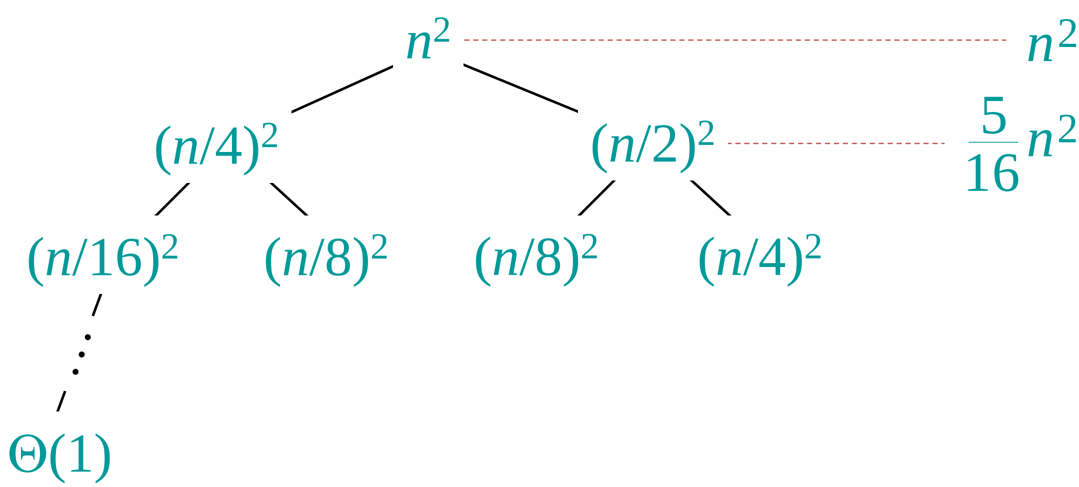
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



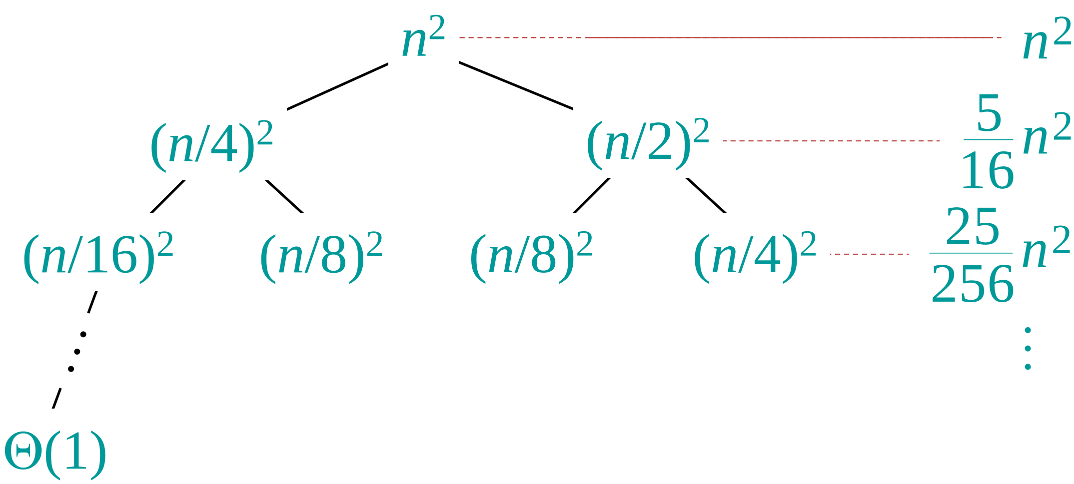
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

n^2 $\quad\cdots\quad$ n^2

$(n/4)^2$ $\quad$ $(n/2)^2$ $\quad\cdots\quad$ $\frac{5}{16} n^2$

$(n/16)^2$ $\quad$ $(n/8)^2$ $\quad$ $(n/8)^2$ $\quad$ $(n/4)^2$ $\quad\cdots\quad$ $\frac{25}{256} n^2$

$\vdots$ $\quad\quad$ $\vdots$

$\Theta(1)$

$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

$$= \Theta(n^2)$$ *geometric series*

12

The master theorem

- Particular case:

$$\boxed{T(n)=\begin{cases} c & n=1 \\ aT\left(\frac{n}{b}\right)+cn & n>1 \end{cases}}$$

- Solution:

$$T(n)=\begin{cases} \Theta(n) & a<b \\ \Theta(n\log_b n) & a=b \\ \Theta\left(n^{\log_b a}\right) & a>b \end{cases}$$

13

The Master Theorem

- if $T(n) = aT(n/b) + f(n)$ where $a \geq 1$ and $b > 1$

then

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & \text{if} \quad f(n) = O\left(n^{\log_b a - \varepsilon}\right) \\ \Theta\left(n^{\log_b a} \log n\right) & \text{if} \quad f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & \text{if} \quad f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \text{AND} \\ & \quad a.f(n/b) \leq c.f(n) \text{ for large } n \end{cases} \quad \begin{matrix} \varepsilon > 0 \\ c < 1 \end{matrix}$$

The Master Theorem

- In each of the three cases, we compare the function $f(n)$ with the function $n^{\log_b a}$. The larger of the two functions determines the solution to the recurrence:
  - Case 1: the function $n^{\log_b a}$ is the larger, then the solution is $T(n) = \Theta\left(n^{\log_b a}\right)$
  - Case 2: the two functions are the same size, we multiply by a logarithmic factor, and the solution is $T(n) = \Theta\left(n^{\log_b a} \log n\right) = \Theta\left(f(n) \log n\right)$

15

The Master Theorem

– Case 3:the function $f(n)$ is the larger, then the solution is $T(n)=\Theta(f(n))$

Remark:

- In the first case, not only must $f(n)$ be smaller than $n^{\log_b a}$, it must be polynomially smaller: $f(n)$ must be asymptotically smaller than $n^{\log_b a}$ by a factor of $n^{\varepsilon}, \varepsilon > 0$

The Master Theorem

Note:

- The three cases do not cover all the possibilities for $f(n)$:
  - There is a gap between cases 1 and 2 when $f(n)$ is smaller than $n^{\log_b a}$ but not polynomially smaller.
  - Similarly, there is a gap between cases 2 and 3 when $f(n)$ is larger than $n^{\log_b a}$ but not polynomially larger.
- The Master theorem cannot be used to solve a recurrence if $f(n)$ falls in one of these cases.

Using The Master Method

- $T(n) = 9T(n/3) + n$
  - a=9, b=3, $f(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
  - Since $f(n) = O(n^{\log_3 9 - \varepsilon})$, where ε=1, case 1 applies:

$$T(n) = \Theta\left(n^{\log_b a}\right) \text{when } f(n) = O\left(n^{\log_b a - \varepsilon}\right)$$

  - Thus the solution is $T(n) = \Theta(n^2)$

18

Using The Master Method

- $T(n) = T(2n/3) + 1$
  - a=1, b=3/2, $f(n) = 1$
  - $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
  - Since $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$, case 2 applies
  - Thus the solution is $T(n) = \Theta(\log n)$

19

Using The Master Method

- $T(n) = 3T(n/4) + n \log n$
  - $a=3$, $b=4$, $f(n) = n \log n$
  - $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$
  - Since $f(n) = \Omega(n^{\log_4 3 + \varepsilon})$, where $\varepsilon \approx 0.2$, case 3 applies if we can show that the regularity condition holds for $f(n)$.
  - For sufficiently large n, $af(n/b) = 3(n/4)\log(n/4) <= (3/4)n \log n = cf(n)$ for $c = 3/4$.
  - Thus the solution is $T(n) = \Theta(n \log n)$

Using The Master Method

- The master method does not apply to the recurrence:
  $T(n) = 2T(n/2) + n \log n$
  - a=2, b=2, $f(n) = n \log n$
  - $n^{\log_b a} = n$
  - Case 3 should apply since $f(n)$ is asymptotically larger than $n^{\log_b a} = n$.
  - The problem is that it is not polynomially larger: The ratio $f(n)/ n^{\log_b a} = (n \log n)/n = \log n$ which is asymptotically less than n^{ε} for any $\varepsilon > 0$.
  - The recurrence falls into the gap between case 2 and case 3.

21

Reading

Chapter 4

Anany Levitin, Introduction to the design and analysis of algorithms, 3rd Edition, Pearson, 2011.